Домашнее задание №6: «Каракули и нейросети»

Дедлайн 1 (20 баллов): 20 апреля, 23:59
Дедлайн 2 (10 баллов): 27 апреля, 23:59

Домашнее задание нужно написать на Python и сдать в виде одного файла. Правило именования файла: `name_surname_6.[py | ipnb]`. Например, если вас зовут Иван Петров, то имя файла должно быть: `ivan_petrov_6.py` или `ivan_petrov_6.ipnb`.

В этом домашнем задании мы продолжим тему распознавания образов. Для тестирования нашего алгоритма будем использовать датасет MNIST [1].

MNIST (Mixed National Institute of Standards and Technology database) является основной базой при тестировании систем распознавания образов, а также широко используемой для обучения и тестирования алгоритмов машинного обучения. Она была создана перегруппировкой образов из оригинальной базы NIST, которая являлась достаточно сложной для распознавания. Кроме этого, были выполнены определенные преобразования (образы были нормализованы и сглажены для получения градаций серого цвета).

1. По ссылке [2] находится датасет для текущего домашнего задания. Данные находятся в формате *.mat, поэтому лучше всего воспользоваться встроенной функцией `scipy.io.loadmat` для чтения данных:

```
import scipy.io
from sklearn.cross_validation import train_test_split

dataset = scipy.io.loadmat('mnist-original.mat')
trainX, testX, trainY, testY = train_test_split(
    dataset['data'].T / 255.0, dataset['label'].squeeze().astype("int0"), test_size = 0.3)
```

[1] http://en.wikipedia.org/wiki/MNIST_database

[2] https://github.com/amplab/datascience-sp14/raw/master/lab7/mldata/mnist-original.mat

2. Перед работой всегда хорошо бы посмотреть на датасет. Для визуализации датасета можно воспользоваться функцией `visualize_mnist` из файла по ссылке. [3]

Картинка должна выглядеть подобным образом:

3. Реализуйте обучение нейронной сети с помощью метода обратного распространения ошибки (Back Propagation).

В качестве функции активации в данном задании допустимо воспользоваться сигмоидой, функцию потерь нужно взять из лекции.

Структура класса приведена ниже:

```
class NeuralNetwork:
    def __init__(self, layers):
        self.num_layers = len(layers)
        self.layers = layers
        ...

    def train(self, X, y, max_iter=10000, learning_rate=1):
        ...
        for j in range(max_iter):
          self.forward(X)
          self.backward(X, y)
          ...

    def forward(self, X):
        ...

    def backward(self, X, y):
        ...

    def predict(self, X):
        ...
```

Параметр конструктора `layers` задаёт количество нейронов в каждом слое в виде списка.

На `forward` шаге объект проходит через нейросеть и вычисляются выходные значения нейронов скрытых слоёв и выходного слоя. На `backward` шаге вычисляются производные, необходимые для обновления массива весов.

При реализации метода `train` может быть полезно обратиться к своей реализации метода стохастического градиента из предыдущего домашнего задания.

4. Дополните реализацию методом `predict`, который прогоняет все объекты из переданной матрицы `X` через обученную нейросеть. Метод должен возвращать вектор, состоящий из индексов нейронов, на котором значение для соответствующего объекта на выходном слое максимально.

[3] `https://gist.github.com/ktisha/95fcee0ed79236c7e6e5`

5. Пример использования полученной сети:

```
nn = NeuralNetwork([train_X.shape[1], 100, 10])
nn.train(trainX, trainY)
nn.predict(testY)
```

Обратите внимание, что на первом слое нам нужно число нейронов, равное количеству признаков (в данном случае, количеству пикселей), а на выходном слое количество нейронов, равное количеству классов объектов (в нашем случае это цифры 0-9).

6. Оценивать качество классификации в этот раз мы будем с помощью простым подсчетом отношения правильно классифицированных объектов к общему количеству объектов в выборке.

7. Ответьте на вопрос: как меняется качество классификации при изменении количества слоев сети и количества нейронов на каждом слое?